# Potential Reduction of Cost for Software Development and Integration Environments

**Sean Barie, Richard Crutchfield**
**MITRE Corporation**
**Orlando, FL**
**sbarie@mitre.org**
**crutchfield@mitre.org**

**Paul Dumanoir**
**U.S. Army PEO STRI**
**Orlando, FL**
**paul.h.dumanoir.civ@mail.mil**

## ABSTRACT

Simulation based training systems are used widely across the Army to train military personnel on a wide array of skillsets. These systems are typically developed, integrated, and maintained by Department of Defense (DoD) contractors. In support of this, the Army pays for the hardware resources required by the contractors for development, integration, and testing purposes. This requires a significant Army investment in the instantiation and maintenance of these hardware environments.

This paper will discuss an approach to potentially reducing these costs by migrating these different environments into the cloud. Along these lines, the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) is currently developing a Live-Synthetic, Training, and Test and Evaluation Enterprise Architecture (LS TTE EA) framework, with the necessary business model, governance structure and reference architecture mechanisms that support a goal of leveraging the cloud to reduce the overall costs required to support training and operational testing. Towards this goal, PEO STRI has developed a Proof of Concept (PoC) implementation of the reference architecture of the LS TTE EA. In this paper, we provide an overview of the LS TTE EA and discuss how it's reference architecture PoC implementation can provision the infrastructure required to support development, integration, and test environments needed for the LS TTE EA community of interest. Using this approach, the Army would no longer be required to pay for additional sets of hardware for each contractor, but instead provision the required resources in the cloud on-demand. When the contractor no longer requires the resources they are reclaimed by the cloud and immediately available for use by others. Since Cloud Service Providers follow the "pay for what you use" paradigm, the Army would only be billed for what is actively being used at a given point in time. Along with describing the cloud based provisioning of these environments, the paper provides a high level cost comparison of how it is done today (purchasing resources) compared to the cost of hosting these environments in the cloud.

## ABOUT THE AUTHORS

**Sean Barie** currently works for the MITRE Corporation as a Lead Software Systems Engineer. He has been part of the U.S. DoD Modeling and Simulation community since 2005 and with MITRE since 2012. He is currently the project leader supporting the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) for the Integrated Training Environment (ITE). He holds a B.S. in Computer Science.

**Paul Dumanoir** is the Chief Engineer for the United States Army Product Manager for Warrior Training Integration (PdM WTI) under the Project Manager for Integrated Training Environment (PM ITE) at PEO STRI. He is currently the PdM WTI Modernization Risk Reduction project manager and has 28+ years of experience working in Army and DoD simulation and training programs as Product Manager, Project Director, and Systems / Software Engineer. His current interests include component-based product-line engineering, enterprise architectures, and system of system integration and interoperability. He earned his B.S. in Electrical Engineering from the University of South Alabama in 1987 and his M.S. in Computer Systems from the University of Central Florida in 1991.

**Richard Crutchfield** currently works for the MITRE Corporation as the Army Training and Learning Technology Group Leader. He has been part of the U.S. DoD M&S community since 1992 and with MITRE since 2007 starting out as a Modeling and Simulation Engineer supporting the Joint Land Component Constructive Training Capability. He has worked as systems engineer, software architect, and software developer on Joint and Army training and

analytical simulations systems including: CSSTSS, FSCATT, WARSIM, JLCCTC, LVC-IA. He holds a B.S. in Mathematics with a concentration in Computer Science.

# Potential Reduction of Cost for Software Development and Integration Environments

**Sean Barie, Richard Crutchfield**
**MITRE Corporation**
**Orlando, FL**
**sbarie@mitre.org**
**crutchfield@mitre.org**

**Paul Dumanoir**
**U.S. Army PEO STRI**
**Orlando, FL**
**paul.h.dumanoir.civ@mail.mil**

## INTRODUCTION

Simulation based training systems are used widely across the Army to train military personnel on a wide array of skillsets. These systems are typically software centric, custom developed applications that have been tailored to provide a particular training capability. They are generally developed, integrated, and maintained by Department of Defense (DoD) contractors. In support of this the Army pays for all hardware resources required by the contractors for development, integration, and testing purposes. This cost is significant when we consider the overall number of DoD contractors supporting the Army.

This paper will provide an overview of the Army's current approach, discuss an alternative approach leveraging the cloud, and lastly provide a comparison of both approaches.

## CURRENT APPROACH

When a contract is awarded to develop a simulation based training system, the cost of the hardware resources required to complete the effort are included in the contractor's bid. These resources are typically hosted at the contractor site and maintained by employees of the contractor at the expense of the Army. While the environments required by a contractor to develop a training system can vary, based on the experience of the authors, we will assume the following environments are required:

1) Software Development Environment
2) Integration and Test Environment
3) Performance Testing Environment

The Software Development Environment is where the software for the training system is developed and maintained. As part of the development process, software developers will also perform some initial testing within this environment to verify the changes they have made are working as intended. Separate from the Software Development Environment, is the Integration and Test Environment. This environment is used to conduct the overall system level integration testing. It is separate from the Software Development environment to ensure it is kept in a pristine state, and also to prevent collisions between development and integration activities. Lastly, a Performance Testing environment is where system wide performance tests are conducted which is separate from the other environments to eliminate any external variables that could influence the test. We will now discuss each of these environments in more detail.

### Software Development Environment

A software development environment is where the software for the training system is developed and maintained. When developing a training system for the Army, it is typically a Controlled Development Environment (CDE). A CDE is an isolated environment separate from corporate resources and from the public internet. This is done as a security mechanism to control the overall access individuals have to the system. The machines that comprise this environment typically fall into one of two possible categories: machines hosting the infrastructure to support development activities and machines used by each individual software developer to develop the software. We will address each of these in the following sections.

### Machines Hosting the Infrastructure to Support Development Activities

While the infrastructure required to support development can vary depending on a number of factors (i.e. development methodology, language) at a minimum it typically consists of the following:

1) Version Control System: Central repository for the software that is being developed. Keeps track of the changes made to individual files.
2) Issue Tracker: Software that allows the management of issues with the software that is being developed. These issues could be defects with the software or new feature requests.
3) Collaboration / Documentation Server: Allows for collaborative documentation of the software being developed.
4) Continuous Integration Server: Allows for automated builds and regression testing of the software being developed.
5) Software Artifact Repository: Allows for the storage of software artifacts (i.e. libraries, executables) that are associated with the software being developed.

Each of these systems are generally given their own dedicated server and installed, configured, and updated manually by contractor personnel.

### Machines used Directly by Software Developers

Since the development environment is generally segregated from corporate infrastructure, each software developer is provided a local workstation where they perform their work. Developers typically have a separate machine provided by their company that has internet access. The development workstation at a minimum will contain the following applications:

1) Integrated Development Environment (IDE)
    a. Software application which normally consists of a source code editor and debugger.
2) Software required to build and run the software components that are being developed.
3) Client to the version control system.

Like the infrastructure to support development, the development workstation is installed, configured, and updated manually by contractor IT support staff. The developer uses this machine directly to make changes to the software components for the training system. As mentioned previously, as part of the overall development process, developers perform some initial testing to verify the changes they have made. In many cases, the simulations being developed are rather resource intensive at runtime, so the developer is also provided a server class machine to support this type of testing. These machine resources are also installed, configured, and updated manually by contractor personnel.

### Integration and Test Environment

As developers develop and test individual components of the overall training system, there comes a point where all the components need to be integrated and tested to verify the components function properly as a whole. In order to test the components together, a separate Integration and Test environments is employed. From a hardware resources perspective this environment will mirror a production environment for the training system.

### Performance Testing Environment

Along with integration testing, performance testing of the overall system is critical to the success of the training system. This is completed in a separate environment from the development or integration environments to ensure no external factors skew the results of the test. Like the Integration and Testing environment, this environment will also mirror a production environment for the training system.

## COLLABORATIVE CLOUD DEVELOPMENT ENVIRONMENT

### Background

The Army is exploring the feasibility of migrating to a cloud based architecture for its simulation based training systems. This includes migrating the environments used for their development, integration, and testing. Some motivating factors behind this are:

1) Compliance with the Common Operating Environment (ASA(ALT), 2013).
2) Joint Information Environment Data Center Mandate (Under Secretary of the Army, 2014).
3) Federal Cloud Computing Strategy (Kundra, 2011)
4) Data Center Reduction Effort (Under Secretary of the Army, 2014)

The Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI) is responsible for the development, acquisition, and sustainment of systems that provide training for the U.S. Army. One of the training system of systems maintained by PEO STRI is the Integrated Training Environment (ITE). The training systems of the ITE are integrated together via the Live, Virtual, and Constructive Integrating Architecture (LVC-IA). LVC-IA combines and connects multiple training systems together in a persistent and consistent manner to effectively train mission essential tasks. Since the ITE is a large system of systems, the effort of migrating it to the cloud is not a small undertaking. As an initial step towards this goal, PEO STRI has developed a Proof of Concept (PoC) Live Synthetic Training and Test and Evaluation Enterprise Architecture (LS TTE EA) (Dumanoir et al, 2015). This PoC aims to provide a clear frame of reference allowing stakeholders from the LS TTE communities to master the linkages between technical architectures and the business and strategic objectives.

**Live Synthetic Training and Test and Evaluation Enterprise Architecture (LS TTE EA)**
The LS TTE EA's goal is to define, align, and evolve a business model, a technical reference architecture, and a governance structure that drives technical and engineering decisions related to Live-Synthetic Interoperability challenges. The reference architecture is a service oriented architecture (SOA) that was based on the Open Group Technical Standard for SOA Reference Architectures (The Open Group, 2011).

**Live Synthetic Training and Test and Evaluation Infrastructure Architecture (LS TTE IA)**
The software architecture PoC implementation of the LS TTE EA is known as the Live Synthetic Training and Test and Evaluation Infrastructure Architecture (LS TTE IA) (Dumanoir et al., 2015). The LS TTE IA is intended to be a framework to support the development and hosting of future cloud based training capabilities. It is also intended to provide a set of reusable architectural structural elements that various solution architectures can reuse to meet their Program of Record (POR) requirements. LS TTE IA also aligns with the Army Common Operating Environment (COE) reference architecture and facilitates hosting within the COE Data Center/Cloud Computing Environment (ASA(ALT), 2013). The high level architectural design of the LS TTE IA PoC is depicted in Figure 1. This LS TTE IA framework is currently being used as an architectural framework for several PM ITE modernization risk reduction efforts, to include the collaborative cloud development environment effort described in this paper. LS TTE IA is composed of six horizontal layers and one vertical layer. These logical layers enable the representation of separation of concerns to achieve a more loosely coupled system (Dumanoir et al., 2015).

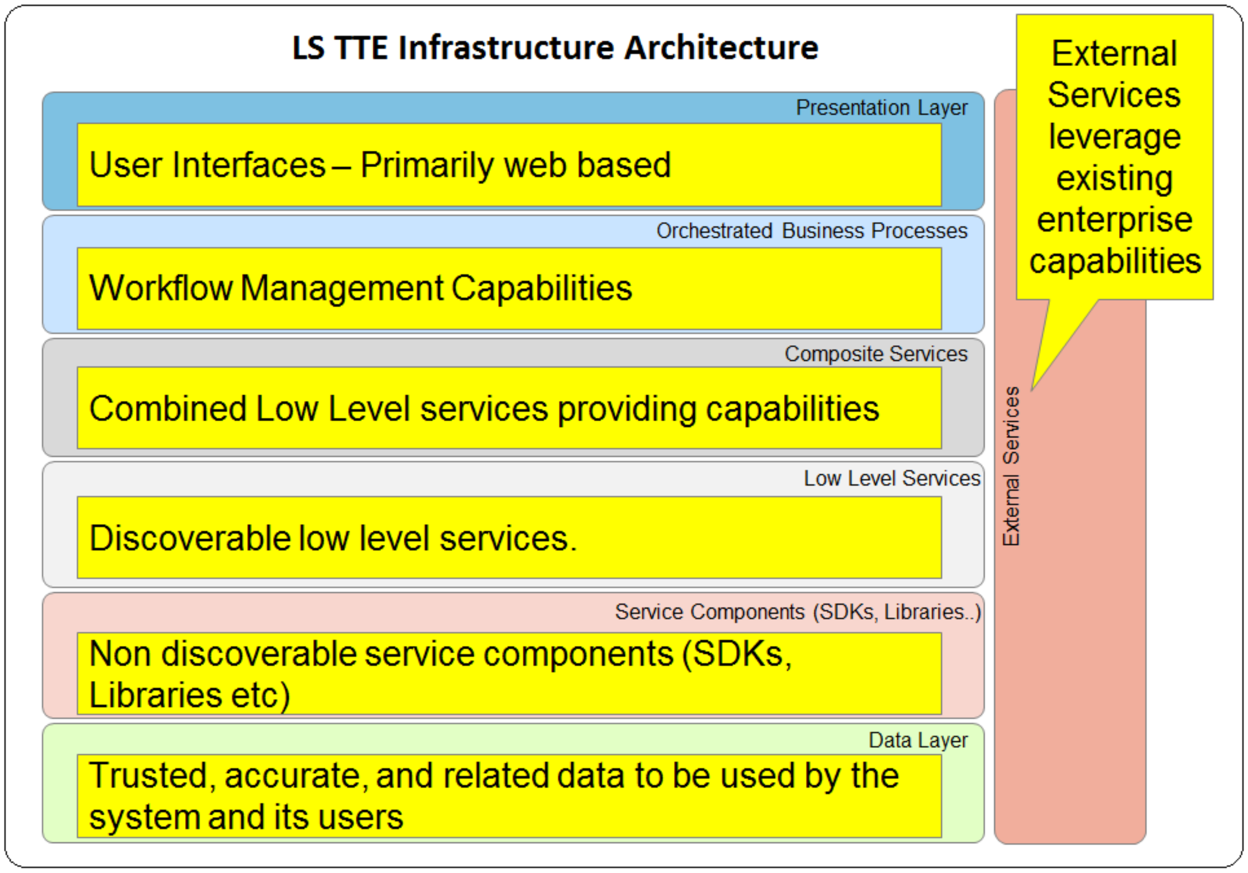
LS TTE Infrastructure Architecture
Presentation Layer
User Interfaces – Primarily web based
Orchestrated Business Processes
Workflow Management Capabilities
Composite Services
Combined Low Level services providing capabilities
Low Level Services
Discoverable low level services.
Service Components (SDKs, Libraries..)
Non discoverable service components (SDKs, Libraries etc)
Data Layer
Trusted, accurate, and related data to be used by the system and its users
External Services
External Services leverage existing enterprise capabilities

**Figure 1. LS TTE IA**

Training systems like the ITE are complicated systems of systems that require highly skilled staff to configure, execute, and teardown. One of the primary objectives of the LS TTE IA is to build upon the success of LVC-IA and further reduce the amount of time and expertise it takes to complete these tasks. It aims to accomplish this by delivering its capabilities using a Software as a Service (SaaS) model (Mell, Grance, 2011) as part of a SOA implementation residing in a cloud based infrastructure. By leveraging open source IT automation tools, LS TTE IA is able to automate today's labor intensive processes like installation and configuration. This has the potential to take a manual error prone process that could take weeks for a large training event, to an error free automated process that completes in tens of minutes. While the technologies exist, implementing this for a complicated system of systems like the ITE still has significant technical challenges. As an initial step towards this goal, a new service has been added to the LS TTE IA prototype: The Computing Environment Service.

**Computing Environment Service**

In order to support the automatic installation and configuration of training systems, the concept of a computing environment (CE) is used. A computing environment is comprised of one or more applications and each application consists of one or more virtual machines (VM). Using this paradigm each instance of a training system can be thought of as an instance of a computing environment. Figure 2 provides a graphical representation of an environment for a training system comprised of two applications.

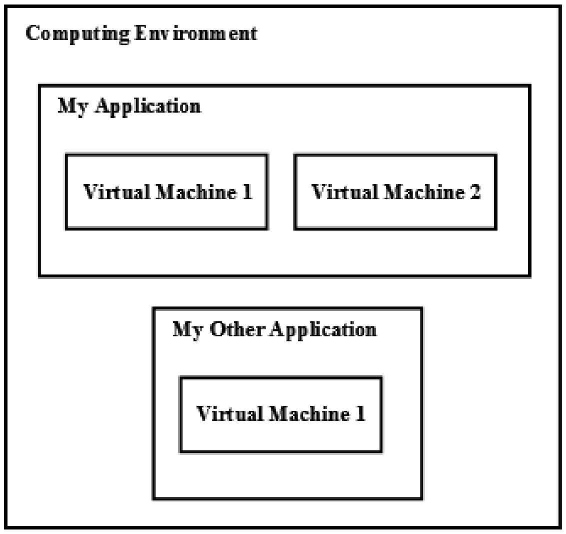

**Figure 2. Computing Environment**

In order to support the definition and automatic provisioning of a CE, two open source tools were leveraged: Terraform and Chef. While this paper will discuss the tools that were chosen, the authors want to emphasize that in this proof of concept, a trade study was not completed. There may be other tools that provide similar capabilities that could perhaps be a better choice. With that being said, the MITRE team chose to use Terraform and Chef.

Chef is an IT Automation tool that supports the automation of application installation, configuration, and management (Chef, n.d.). It does this by allowing a system administrator to create one or more scripts (referred to as recipes) that install and configure a particular application. By installing a Chef client on a machine, and configuring the client to install the appropriate recipes, Chef will automatically execute the Chef scripts and keep track of the overall configuration of the machine. Any updates the system administrator wants to make can be registered with the Chef Server which is responsible for keeping track of all the clients and the changes will be automatically pushed to all of the affected clients. Also, since Chef periodically checks each of the nodes to verify it is in the correct state (as defined by the recipe) it can be used to enforce system wide policies (like Information Assurance for example).

Chef provides a script based approach to installing software on a virtual machine, but in order to provision a CE, LS TTE IA needs a similar capability to create the virtual machine itself. For this purpose, the MITRE team pulled in Terraform (Terraform by HashiCorp. (n.d.)). Terraform allows a system administrator to define a script that specifies the hardware resources required. It supports a wide range of resources including virtual machines. Using this, a script can be defined that details each VM that needs to be created for a particular CE. Terraform is compatible with Chef, so as part of this specification the Chef recipes that need to be applied to the VM can also be specified. Lastly, since it is still unclear exactly which Cloud Service Provider (CSP) the Army will move to, terraform provides a layer of abstraction from the CSP itself (supports many CSPs).

By using a combination of Terraform and Chef scripts, the VMs and applications that make up a CE can now be defined. In other words, a Terraform script can be created detailing each VM and its requirements (i.e. CPU, memory) that is required for the CE. Chef scripts can also be created and referenced via the Terraform scripts to install the required software on each VM. By taking these scripts as inputs, the Computing Environment Service can define and automatically provision a computing environment without knowing all the details of the underlying system.

## Collaborative Cloud Development Environment Proof of Concept

In an effort to determine its overall feasibility, and also to support the development of the LS TTE IA, a proof of concept (PoC) Collaborative Cloud Development Environment (CCDE) is being created. The goal of this effort is to explore the viability of developing, integrating, and testing DoD simulation based training systems in the cloud. As an initial step towards this goal, PEO STRI in cooperation with MITRE, has put together a phased approach to implementing a PoC Collaborative Cloud Development Environment. The initial phase, expected to be completed by the end of FY16, involves manually standing up a development environment for the LS TTE IA in Amazon GovCloud

(Amazon Web Services (AWS) – Cloud Computing Services, n.d.). A potential follow on phase in FY17 will explore using an instance of the LS TTE IA (hosted in Amazon GovCloud) to automatically provision software development, integration and testing, and performance testing environments. This could be used to not only provision these environments for the LS TTE IA itself, but also for use by contractors developing a capability for use on the LS TTE IA framework.

While the LS TTE IA is still a proof of concept implementation, contractors are already prototyping services on it to explore other areas of interest for PEO STRI. Currently these contractors are forced to complete development locally and lack a convenient place to test their developed software against the LS TTE IA. They also must be provided the source code and documentation for the LS TTE IA on a point to point basis, instead of just accessing it from a shared repository. The PoC CCDE would allow an instance of the LS TTE IA to be hosted in the cloud and available for integration and testing purposes by authenticated users. It would also host the source code for LS TTE IA to better support external contributions from contractors with the appropriate credentials. To support the development of their services, the contractors would also be provided a cloud based development environment. Like what was done in the current approach, the next sections will discuss this approach in more detail for each environment. While the following sections focus on contractors supporting the LS TTE IA, the PoC CCDE may be applicable to contractors supporting the DoD in general.

**Software Development Environment**
The PoC CCDE would provide a contractor, tasked with developing a capability using the LS TTE IA framework, access to a web page that would walk them through the process of creating their environments in the cloud. This web page would be provided by the LS TTE IA and would leverage the Computing Environment Service to create the environments. Using this web page, the contractor would begin the process of creating a cloud based software development environment. As part of creating this environment, the contractor would be given a list of applications to select from. These selected applications/services would automatically be included in the newly created environment. The following types of systems would be provided for selection:

1) Version Control System
2) Issue Tracker
3) Collaboration / Documentation Server
4) Continuous Integration Server
5) Software Artifact Repository
6) LS TTE IA Services

For each of these applications/services, the resources required to support them (i.e. virtual machines, software) have been codified ahead of time using Terraform and Chef. Once the contractor selects the applications to include in their environment, the Computing Environment Service will execute the Terraform and Chef scripts for each application to automatically build out the environment. Upon completion, the contractor can access the created systems and begin the software development process. CSP provided authentication mechanisms (typically public/private key based) would ensure only the contractor has access to these systems. As part of the selection process, the contractor could also select LS TTE IA services to be included in the environment. This may include services like the Simple Storage Service, SOA Registry, or even their own instance of a Computing Environment Service in case they want to provision additional resources within their environment (Additional machines for limited I&T testing for example).

Along with the infrastructure to support software development, development machines for each developer could also be provisioned as part of the environment creation process. A Virtual Desktop Infrastructure (VDI) based solution could be explored to provide access via thin clients to each of the provisioned systems. Having said that, it may be the case the contractor prefers to continue to have developers work using local development machines (many developers prefer this approach). If this is the case, thin clients would not be provided and machines provided by the contractor would be used instead.

**Integration and Test Environment**
Using the same web based wizard provided by LS TTE IA, the next step for the contractor is to create an environment they can use for integration and testing purposes. As mentioned in the current approach, this environment is used for system wide integration testing and is kept separate from the software development environment to ensure it is kept in a pristine state. Like what was done for the creation of the software development environment, once again the

contractor will be provided a list of applications to select from to be automatically included within the environment. One of these services will be the Computing Environment Service. This service will be used by the contractor to build out the capabilities they want to test since, especially in the case of a new capability, the exact resources required to support it may not be known up front.

**Performance Testing Environment**
Similar to the other environments, the web based wizard provided by LS TTE IA will also be used to create an environment for performance testing. As mentioned in the current approach, this environment is used for system wide performance testing and is kept separate from the other environments to eliminate any external variables that could influence testing. Once again the contractor will be provided a list of applications to select from to be automatically included within the environment. Included in this list will be tools to assist with the Performance Testing process (like Gatling for example (Gatling Project, n.d.)). The Computing Environment service will also be included so it can be used by the contractor to build out the capabilities they want to test.

## COMPARISON OF APPROACHES

The following sections compare the current approach to the alternative approach of using the PoC CCDE. It does this using several criteria and attempts to answer some of the questions associated with each.

1) Flexibility – How flexible is each approach? How hard is to grow or shrink the environments with the ebbs and flow of development?
2) Security – Which approach is more secure?
3) Cultural Impact – How does this impact the way things are done today?
4) Cost – Is there a potential for cost savings by migrating to the cloud?

### Flexibility

It is the opinion of the authors that hosting the software development, integration and test, and performance testing environments in the cloud offers greater flexibility than the approach of purchasing hardware directly. For example, if the runtime requirements of the system were underestimated, or due to new requirements the training system uses more resources than originally planned, the current approach would force the contractor to go out and purchase additional, or replacement hardware. With a cloud based approach additional resources can be added to the already existing systems. This can greatly reduce the time required to adjust resources and also possibly reduce the associated costs. This concept also applies when we consider the overall ebb and flow of a software development team. New developers will join the team and others will move on to other efforts. Adding and removing resources in support of this is simplified with a cloud based approach.

Another example of additional flexibility is the transfer of these environments when a contract is being handed over to another contractor. With a cloud based solution it could be as simple as transferring the existing environments to the new contractor. With the hardware based solution, either new hardware would need to be purchased and provided, or the original hardware would need to be boxed up and shipped over to the new contractor.

### Security

A locally hosted Controlled Development Environment with no external access greatly reduces the vulnerability of the system from external sources. With a cloud based approach, even with features like Virtual Private Clouds and access via VPN, the system is still accessible remotely and therefore more vulnerable. In addition to this, many public CSPs are not cleared to host classified information. Therefore, if the development of a training system needs access to classified information, a standalone system in a cleared environment would still be required. One area where a cloud based approach may offer some benefits is the implementation of the required security infrastructure. Under the cloud approach, the CSP would be responsible for the establishment and maintenance of the needed firewalls, encryption devices, and authentication/authorization services. Whereas with the current approach, assuming the environment is accessible remotely, this responsibility would fall on the contractor.

### Cultural Impact

One of the most significant challenges facing the concept of a Collaborative Cloud Development Environment is its cultural impact. The idea of hosting intellectual property on an external system will likely encounter great resistance from DoD contractors. In a world where their intellectual property can easily mean success or failure, contractors go to extensive lengths to ensure the proper safe guards are put in place to protect it. Trusting an external Cloud Service Provider with this information will be a significant paradigm shift, and potentially, a non-starter. This impact will also be felt on the government side. Migrating to the CCDE means the source code and potentially even the training systems themselves (for testing purposes) would be hosted on an external system. With external entities actively trying to hack into government systems, relinquishing some control in this area will likely face resistance.

**Cost**

Cost is one of the main reasons for the popularity of cloud computing in today's world. With the ability to have access to essentially an unlimited pool of resources, yet only be charged for what you actually use, cloud computing simplifies and can potentially reduce the costs associated with hardware resources. By leveraging cloud based hardware, the need for IT professionals and local server rooms is greatly reduced or even potentially eliminated. With the "pay as you go" costing approach you are only charged for the hardware resources when you actually use them. For example, in the case of Virtual Machines, instead of purchasing the hardware required to support what is believed to be the maximum number of VMs required, you only pay for the VMs you are actively using. This same concept applies to things like storage. Instead of purchasing a very large storage capability to ensure it meets all of your storage needs, you are only billed for the storage you actually use.

However, all of the flexibility associated with cost for a cloud based solution can also have its downsides. With a cloud based approach controls would need to be put in place to ensure the cloud is used in a cost effective manner. It would be rather trivial for a contractor to use the cloud inefficiently and drastically drive up the costs for the Army. For example, a contractor could build an excessive number of environments that aren't required and drive up the overall costs of the system. To prevent this, limits would need to be put in place that only allow a contractor to consume a certain amount of resources. To go over that amount, the contractor would need to contact the government for approval.

**Software Development and Integration and Test Environments**

To further explore the costs associated with both approaches, we will consider the establishment and usage of software development and integration and testing environments to perform a high level cost comparison. Since from a hardware perspective, the performance testing environment is very similar to the integration and testing environment, we will omit this environment from the comparison. Also, we have limited the scope of the comparison to only the servers required for each environment. Lastly, this comparison is not intended to act as proof that a cloud based solution is more cost effective, but instead aims to show that the potential for cost savings does exist. A much more comprehensive and robust cost comparison will need to be completed to ensure a cloud based approach is indeed more cost effective.

For costing purposes Dell's website (Dell, n.d.) was used to obtain prices for the current approach and Amazon Web Service's (Amazon Web Services (AWS) – Cloud Computing Services, n.d.) website was used to obtain costs for the cloud based approach. Due to differences in the hardware provided by each, an exact match could not be made for the servers selected. Because of this, the closest hardware available from each was selected. In order to make the costing data easier to follow the servers have been categorized into two categories: Low End Server and High End Server. Table 1 provides the specifications for each category by approach.

**Table 1. Server Category Definitions**

| | **Current Approach** | **Cloud Based Approach** |
|---|---|---|
| **Low End Server** | Intel® Xeon® E5-2623 v3 3.0GHz<br>4 cores - 12 GB RAM | Intel Xeon E5-2666 v3 2.9 GHz<br>4 cores - 15GB RAM |
| **High End Server** | Intel® Xeon® E5-2640 v3 2.6GHz<br>8 cores - 24 GB RAM | Intel Xeon E5-2666 v3 2.9 GHz<br>8 cores - 30GB RAM |

With the server categories defined the following assumptions were made in regards to the environments themselves:

1) 5 high end servers are required to run the training system.
2) The software development environment requires the following:
    a. 5 low end servers for hosting the infrastructure required for development.
    b. 5 high end servers for developers to test changes.
3) Integration and Test Environment requires the following:
    a. 5 high end server class machines.

Using the above assumptions and the defined server categories, Table 2 below shows the combined costs for both environments over a one-year period of performance (POP) for the current approach.

**Table 2. Current Approach Costs – 1 Year POP**

| | **Quantity/Time** | **Cost** | **Total** |
|---|---|---|---|
| **Low End Servers** | 5 servers | $2,301.76 each | $11,508.80 |
| **High End Servers** | 10 servers | $2,880.57 each | $28,805.70 |
| **Setup and Installation** | 60 hours | $89.31 per hour | $5,358.60 |
| **Maintenance** | 360 hours | $89.31 per hour | $32,151.60 |
| **Total** | | | $77,824.70 |

The costs for the servers were taken directly from Dell by creating a system with specifications defined for each category. For the labor costs associated with the setup and maintenance of the environments, the hourly rate of a Senior Network Administrator was used (ManTech, 2015). For setup and installation, it was assumed it would take 4 hours per machine. Maintenance includes manually applying patches and updates, along with resolving any issues that should arise. For this it was assumed 2 hours per machine per month. Table 3 below shows the combined costs for both environments over a one-year POP for the cloud based approach.

**Table 3. Cloud Based Approach Costs – 1 Year POP**

| | **Quantity/Time** | **Cost** | **Total** |
|---|---|---|---|
| **Low End Servers** | 5 servers (50 hrs a week) | $450.45 / month | $5,405.40 |
| **High End Servers** | 10 servers (50 hrs a week) | $1801.70 / month | $21,620.40 |
| **Setup and Installation** | 1 hour | $89.31 / hour | $89.31 |
| **Maintenance** | 96 hours | $89.31 / hour | $8,573.76 |
| **Bandwidth** | 5 TB / month | $460.71 / month | $5,528.52 |
| **Total** | | | $41,217.39 |

All costs for the cloud based approach were obtained using Amazon Web Service's (AWS) Simple Monthly Calculator (Amazon Web Services (AWS) – Cloud Computing Services, n.d.). For both categories of servers, it was assumed they would be available on-demand and used on average 50 hours a week. Since AWS only charges when the resources are being actively used, this lowered the overall costs for the servers. Similar to the current approach, the hourly rate of a Senior Network Administrator was used for calculating the labor costs. By using the web based wizard provided by LS TTE IA to create the environments, the setup and installation time was reduced to an hour. Maintenance time was also reduced with the assumption that an IT Automation tool like Chef is used to automate the process for applying updates and patches. This means instead of having to manually install updates on each machine, a chef script can be created once and simply executed on each of the machines. Bandwidth was added as an additional cost as this is required by AWS. It was assumed the environments would average 5 TB per month of bandwidth for data coming out of the cloud (data going into the cloud was no cost).

## CONCLUSION

In an effort to determine its overall feasibility, and also to support the development of the LS TTE IA, PEO STRI is creating a proof of concept Collaborative Cloud Development Environment. The goal of this effort is to explore the viability of developing, integrating, and testing DoD simulation based training systems in the cloud. While still early on in its development, it aims to explore not only the cost and technical feasibility of migration to a CCDE, but also the potential cultural and security implications associated with such an approach. As a first step, a development environment for the LS TTE IA has been established using Amazon Web Services. This provided some initial insight in regards to the costing paradigm and security posture required for such an approach. In order to further explore its overall feasibility, a pilot should be considered in which the development of an existing system is migrated to use this approach.

## REFERENCES

Amazon Web Services (AWS) – Cloud Computing Services. (n.d.). Retrieved June 15, 2016 from https://aws.amazon.com/

ASA(ALT). (2013). Common Operating Environment Data Center/Cloud Computing Environment Architecture Compliance. Version 2.0.

Chef. (n.d.). Retrieved June 15, 2016 from https://www.chef.io/

Gatling Project. (n.d.). Retrieved June 15, 2016, from http://gatling.io/

Dell. (n.d.). Retrieved June 15, 2016, from http://www.dell.com

Dumanoir, P., Willoughby, M., Grippin, B., Crutchfield, R., Wittman, R., & Barie, S. (2015). *Live Synthetic Training and Test & Evaluation Infrastructure Architecture* (Tech.). Orlando, FL: Interservice/Industry Training, Simulation, and Education Conference.

Kundra, V. (2011). *Federal cloud computing strategy*. Washington: The White House.

Live Training Transformation (LT2). (2013). Objective Architecture Vision. Revised February 12, 2013

Mantech. (2015). *General Purpose Commercial Information Technology Equipment, Software and Services* (Rep.). Fairfax, VA: Mantech.

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. Gaithersburg, MD: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

Terraform by HashiCorp. (n.d.) June 15, 2016, from http://www.terraform.io/

The Open Group. (2011). SOA Reference Architecture. Technical Standard (C119).

Under Secretary of the Army, (2014). Migration of Army Enterprise Systems/Applications to Core Data Centers [Memorandum]. Department of the Army.